SUBMITTED BY: Pranay Mayal, B2 (NON HONS.)

Lab Exercise 8– Creating a VPC in Terraform Objective:

Objective:
Learn how to use Terraform to create a basic Virtual Private Cloud (VPC) in AWS.
Prerequisites:

- Terraform installed on your machine.
- AWS CLI configured with the necessary credentials.

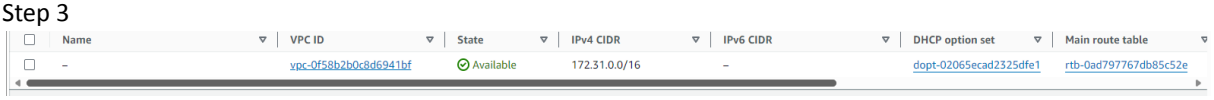
Step 1 Create main.tf

```
provider "aws" {
region = "us-east-1"
 access_key = "AKIAUUSTJKS7TPQGW7CV"
 secret_key = "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"
}
resource "aws_vpc" "my_vpc" {
cidr_block = "10.0.0.0/16"
enable_dns_support = true
enable_dns_hostnames = true
tags = {
Name = "MyVPC"
  }
}

resource "aws_subnet" "my_subnet" {
  count = 2

  vpc_id                  = aws_vpc.my_vpc.id
  cidr_block              = "10.0.${count.index + 1}.0/24"
  availability_zone       = "us-east-1a"
  map_public_ip_on_launch = true

  tags = {
    Name = "MySubnet-${count.index + 1}"
  }
}
```

Step 2

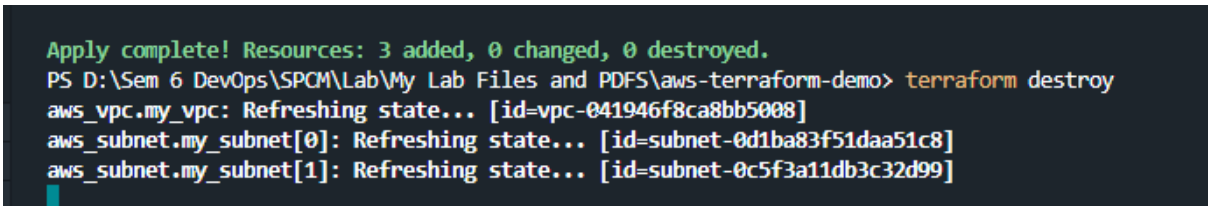
```
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.31.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_subnet.my_subnet[0] will be created
  + resource "aws_subnet" "my_subnet" {
      + arn                                            = (known after apply)
      + assign_ipv6_address_on_creation                = false
      + availability_zone                              = "us-east-1a"
      + availability_zone_id                           = (known after apply)
      + cidr_block                                     = "10.0.1.0/24"
      + enable_dns64                                   = false
      + enable_resource_name_dns_a_record_on_launch    = false
      + enable_resource_name_dns_aaaa_record_on_launch = false
      + id                                             = (known after apply)
      + ipv6_cidr_block_association_id                 = (known after apply)
      + ipv6_native                                    = false
      + map_public_ip_on_launch                        = true
      + owner_id                                       = (known after apply)
```

Step 3



Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP option set	Main route table
–	vpc-0f58b2b0c8d6941bf	Available	172.31.0.0/16	–	dopt-02065ecad2325dfe1	rtb-0ad797767db85c52e

Step 4



```
Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
PS D:\Sem 6 DevOps\SPCM\Lab\My Lab Files and PDFS\aws-terraform-demo> terraform destroy
aws_vpc.my_vpc: Refreshing state... [id=vpc-041946f8ca8bb5008]
aws_subnet.my_subnet[0]: Refreshing state... [id=subnet-0d1ba83f51daa51c8]
aws_subnet.my_subnet[1]: Refreshing state... [id=subnet-0c5f3a11db3c32d99]
```